Middleware Evaluation and Benchmarking for Mission Operations Centers

Ground System Architecture Workshop

March 3, 2005

Rob Antonucci
Emergent Space Technologies

Waka A. Waktola
NASA Goddard Space Flight Center

Outline

- **GMSEC Project and Middleware**
- **Middleware Performance Study**
  - Study Goals and Approach
  - Findings
  - Middleware Perceptions
- **Key Design Considerations**

Goddard Mission Services Evolution Center (GMSEC)

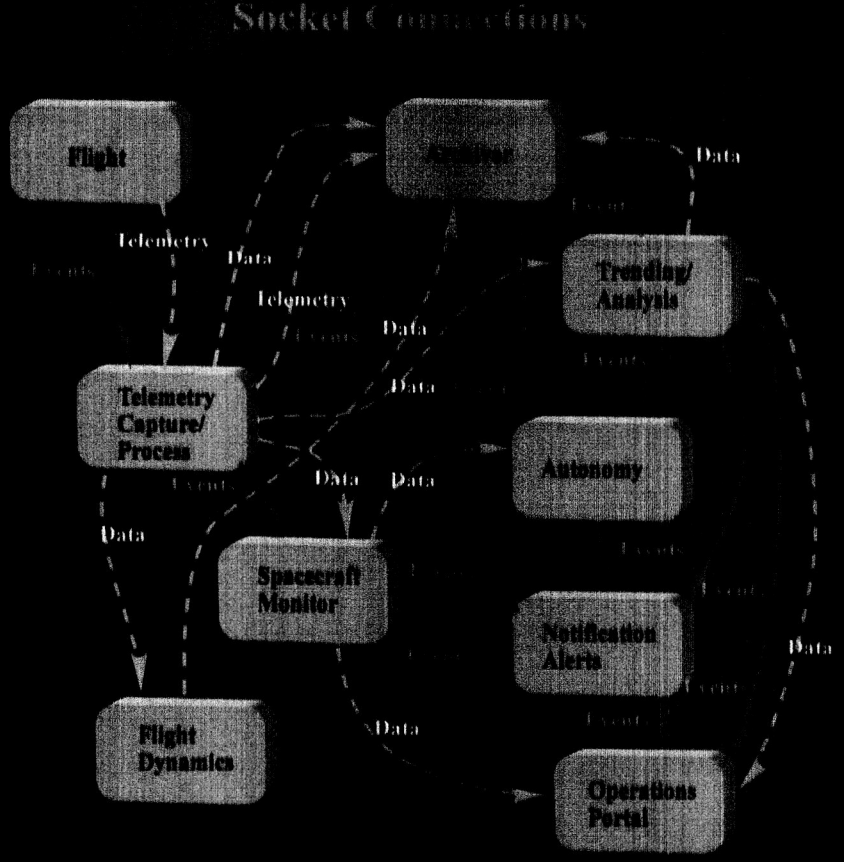
- **Next generation architecture to provide flexible and cost-effective mission services to meet GSFC mission needs**
  - Simplified integration of ground and flight software components
  - Support for evolving operational requirements
  - Simplified infusion of new technologies and components
- **Architecture must have core capability to add, swap and reconfigure individual software components without impact to remaining architecture**
- **Key strategy in meeting that capability is the reliance on middleware for communication and data requirements**

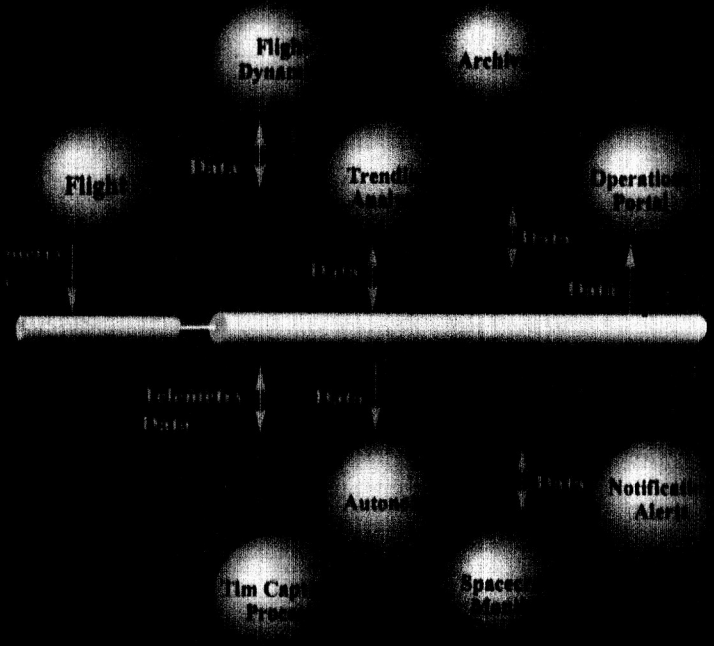
Middleware

- **Middleware is software**
  - Manages communication between disparate applications across heterogeneous computing platforms
  - Routes data transparently between different back-end data sources and end-user applications
- **Decouples data producers and consumers**
  - Easier to add or swap out components
  - Reduces complexity
  - Focus more on data communicated and less on which component application is used

NASA

Middleware in GMSEC Domain





Middleware Performance Study

- **Performance study conducted in 2004**
  - Evaluate and assess candidate middleware products
  - Compare/contrast middleware with point-to-point solutions
  - Validate/refute commonly held perceptions regarding viability of middleware solutions
- **Study performed in two phases**
  - Benchmarking to provide statistical metrics
  - Mission Operations Center (MOC) simulation to provide more realistic operational sanity check

Performance Study Approach

- **Benchmarking**
  - Set of clients producing and consuming generic data across the middleware
  - Tight monitoring of all data transmissions
- **MOC Simulation**
  - Replication of ground station environment with middleware, T&C, and event-analysis components ingesting mission data
- **Target specific areas of assessment**
  - Typical MOC Traffic – *Delay, Reliability*
  - Varied Configurations – *Notable performance changes*
- **Addressing middleware perceptions**
  - Overhead, Guaranteed Messages, Plug and Play, Cost

Candidate Middleware Products

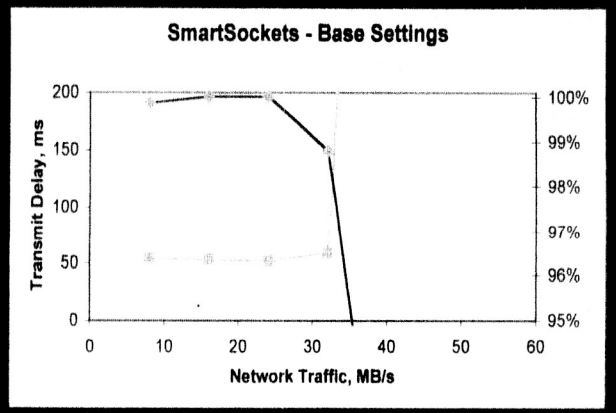
- **TIBCO Smartsockets**
- **TIBCO Rendezvous**
- **ICS Software Bus**
- **Mantara Elvin***
- **IBM Websphere***

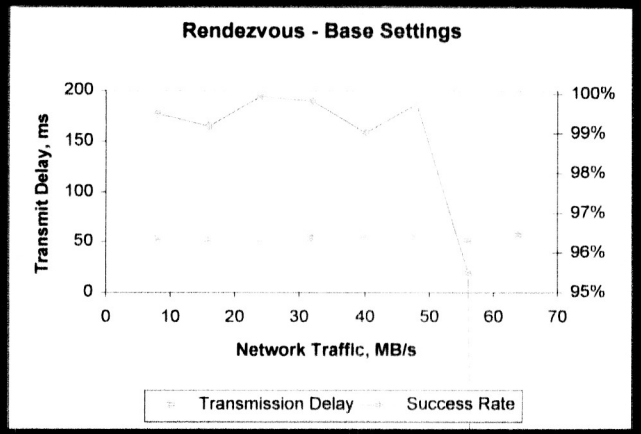
* Evaluated but not yet tested

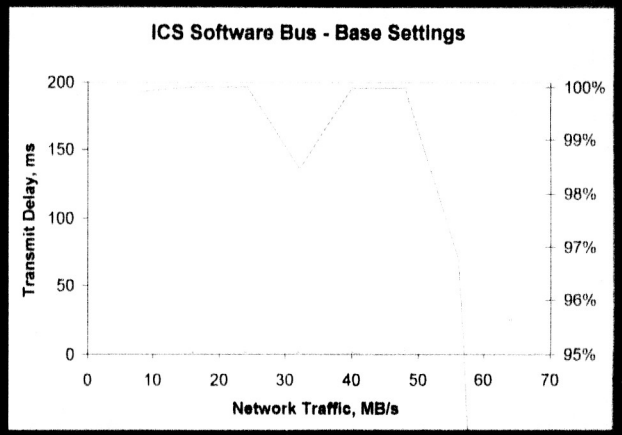
NASA

General Findings: Typical MOC Traffic

- **Performance requirements:**
  - < 100 ms transmission delay
  - > 99% success rate
  - For loads 0-10 Mbps
- **All products met requirements**
- **MOC simulation showed no errors or stress on system for any middleware**

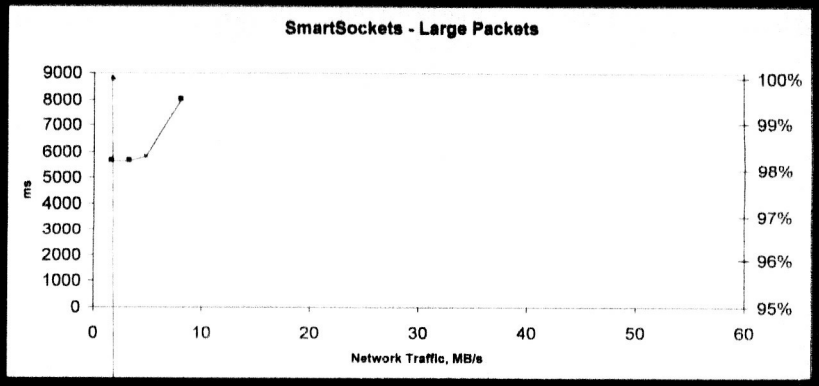


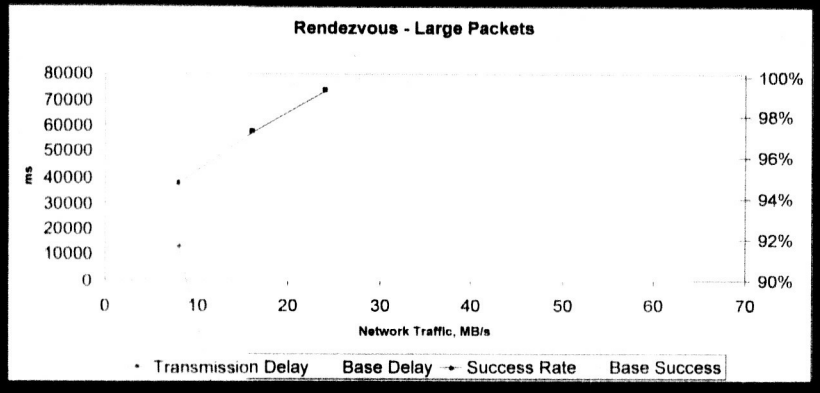


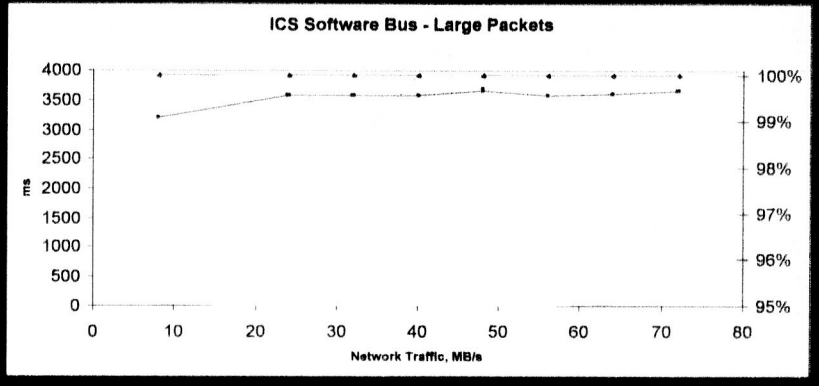


Notable Findings: Configuration I

- **Large Messages**
  - Tested 5MB messages
  - Delay requirement relaxed to 5 sec
  - Only ICS met requirements





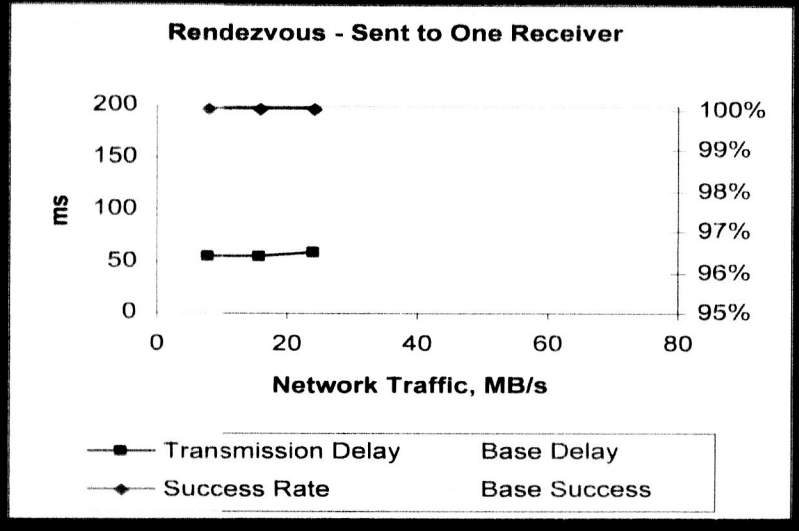


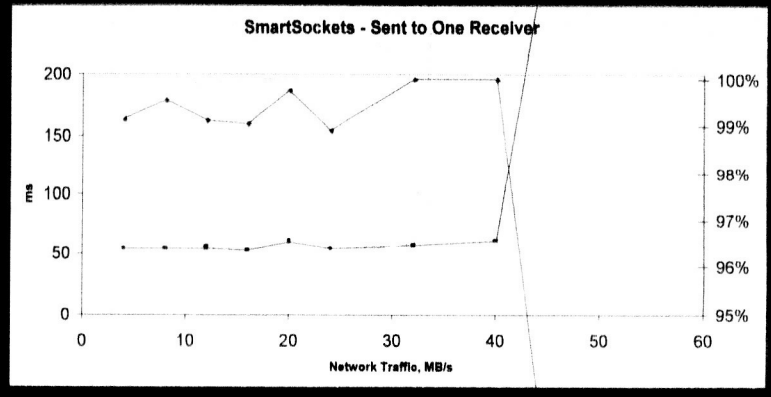
NASA

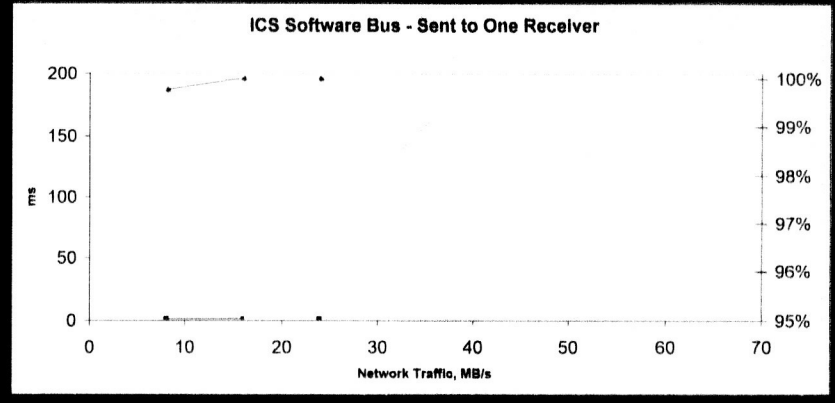
Notable Findings: Configuration II

One-on-One Conversations

- Tested with all traffic from one-on-one conversations
- All products met requirements, but ICS and Rendezvous had noticeably poorer performance







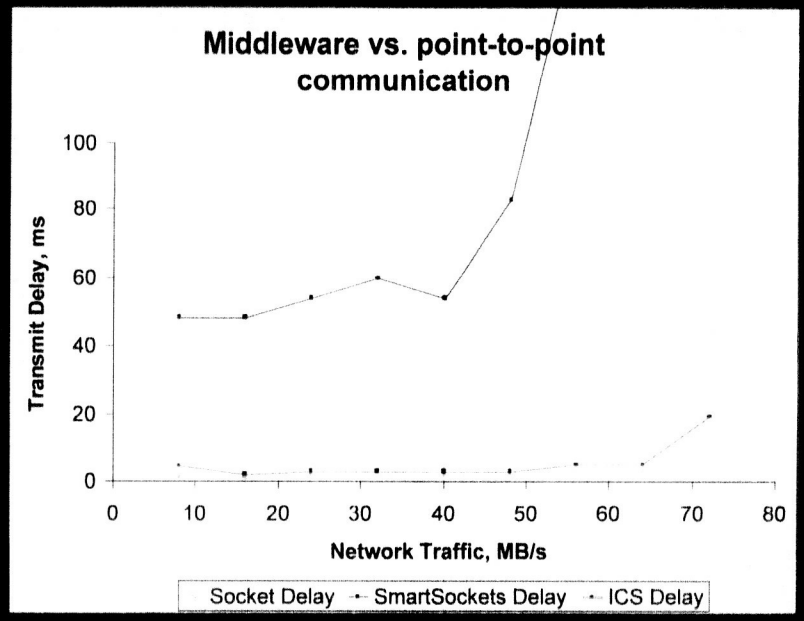
Middleware Perception vs. Reality: Overhead

Perception:

Middleware will impose significant time and throughput overhead

Reality:

Time impact negligible and throughput still exceeds mission needs



Middleware Perception vs. Reality:
Guaranteed Messages

Perception:

- Guaranteed messages does not mean all messages will be successful

Reality:

- Client will be informed if message is not successful
- Extra effort can ensure that message is delivered
  - Point-to-point confirmation for regular messages
  - End-to-end confirmation added for guaranteed messages
  - Clients may have their own end-to-end confirmation mechanism (request/response)
- Messages cached to disk will survive crash

Middleware Perception vs. Reality: Plug and Play

Perception:

- Middleware is not instant interoperability

Reality:

- Middleware can be used in plug-and-play architecture
- Components must talk to middleware
  - Requires recoding of component
  - Bridging applications often used
  - GMSEC API standardizes middleware interface
- Components must understand data
  - GMSEC API provides common data model

Middleware Perception vs. Reality: Cost

Perception:

Middleware solutions make architecture cost-prohibitive

Reality:

There is wide cost variation among middleware products

Required capabilities may need to be closely examined to find best fit

	SmartSockets	Rendezvous	WebSphere	Elvin	ICS
Fault Tolerance	Server + Client	Server + Client	Server + Client	Server	No
Load Balancing	Server + Client	Server + Client	Server + Client	Server	No
Guaranteed Messages	Yes	Yes	Yes	Yes	No
Cost	$45K	$45K	$35K	$15K	Free

Key Design Considerations

- When should middleware be used?
- Should messages be guaranteed?
- Should middleware servers be redundant?
- Should components use middleware redundancy?
- What other characteristics should be considered?

When Should Middleware Be Used?

Pros

Easy to add or swap out components

"Menu" of GMSEC components

Less integration time

Cons

Existing components must migrate

May require development

COTS middleware mandate upgrades

Best For

New missions

Long lived missions

Low budget missions

Missions with changing requirements

Worst For

Existing missions with no need for reengineering

Should Messages Be Guaranteed?

Pros

More reliable

Removes single point of failure

Sender can react if never received

Cons

Poorer performance

May be repeating effort of client

May not want messages to survive crash

Best For

Critical messages

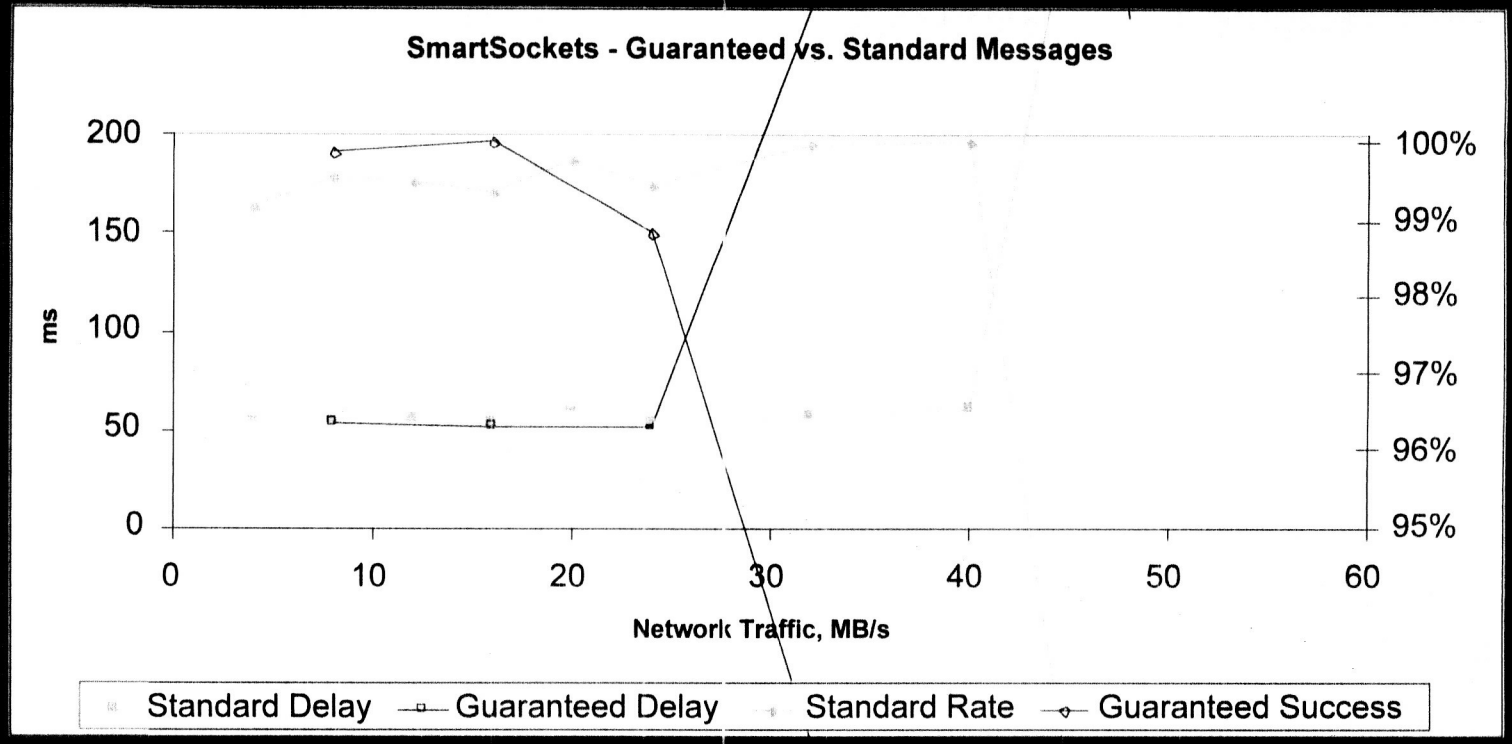
Messages that sender can react to if never received

Worst For

Time sensitive information

High frequency information

Impact of Guaranteed Messages



Too many guaranteed messages actually reduce success rate.

Should Servers Be Redundant?

Pros

Backup server incase primary fails

Removes single point of failure

Failover may be seamless

No change in code required

Best For

Autonomy that cannot support a single point of failure

Cons

Redundancy not offered in less expensive products

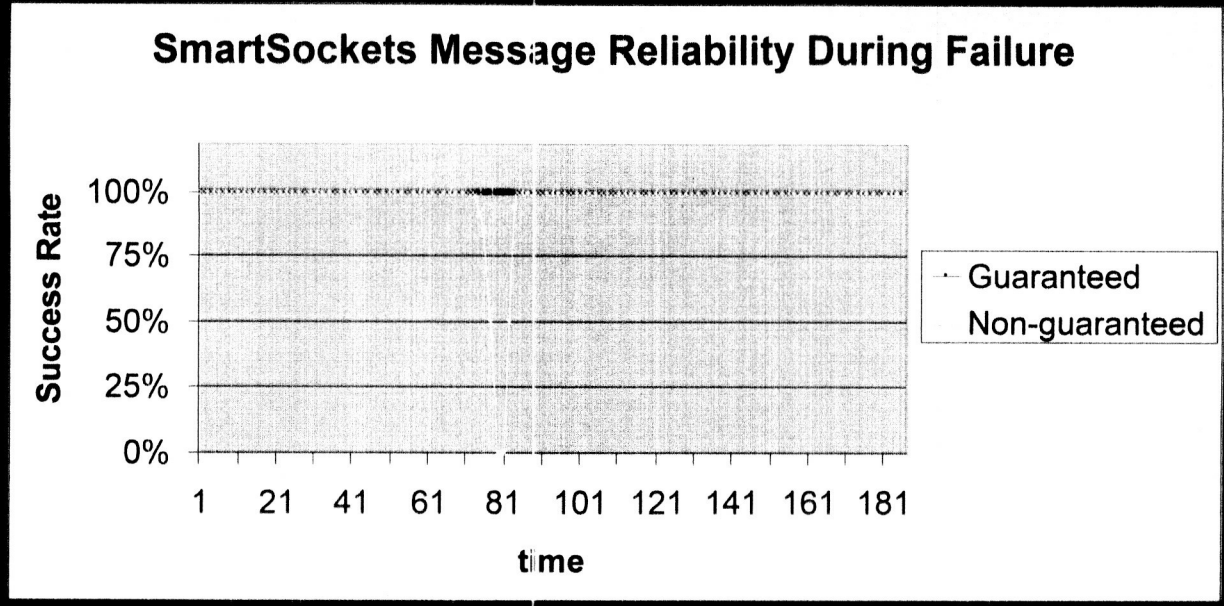
Backup sits idle until needed

Worst For

Low cost missions

Crude autonomy that always pages operators

Impact of Redundant Servers



Redundant servers can still lose messages during a failure.

Should components use middleware redundancy?

Pros

Backup component runs silently until primary fails

Removes single point of failure

No change in code required

Cons

Component redundancy only offered in more expensive products

Best For

Critical components

Worst For

Components with communications outside of the middleware

Components with persistent stores

What Other Characteristics Should Be Considered?

- **What if the expected load exceeds benchmark limits?**
  - Some middleware supports load balancing
  - Multiple servers splitting load
  - Multiple instances of clients with messages delivered round-robin
- **What if the mission has very large data packets?**
  - If middleware does not support very large messages, packets can easily be broken into many smaller messages

Summary

- Middlewares are capable of performing in a mission operational environment
- Cost-effective middleware solutions available for small missions
- Middleware-based architectures are flexible to support evolving mission requirements

Acronyms

API	Applications Programming Interface
COTS	Commercial-Off-The-Shelf
GMSEC	GSFC Mission Services Evolution Center
GSFC	Goddard Space Flight Center
ICS	Interface & Control Systems, Inc.
T&C	Telemetry and Command